

Failure-Tolerant Overlay Trees for Large-Scale Dynamic Networks

Davide Frey¹ and Amy L. Murphy²

¹INRIA Rennes-Bretagne Atlantique, Rennes, France,

`davide.frey@irisa.fr`

²FBK-IRST, Italy, `murphy@fbk.eu`

P2P 2008

Tree vs Mesh

Mesh Topologies

- Inherently Robust

Tree vs Mesh

Mesh Topologies

- Inherently Robust
- Potentially high bandwidth Usage
- Build on-demand trees: latency

Tree vs Mesh

Mesh Topologies

- Inherently Robust
- Potentially high bandwidth Usage
- Build on-demand trees: latency

Trees

- Inherently Fragile
- Cost-Effective Data Dissemination

Tree vs Mesh

Mesh Topologies

- Inherently Robust
- Potentially high bandwidth Usage
- Build on-demand trees: latency

Trees

- Inherently Fragile
- Cost-Effective Data Dissemination

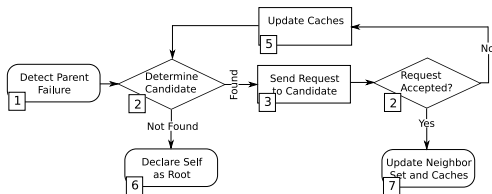
Goals

- Make trees robust: churn resistant
- Control node degree
- Limit impact of topology changes

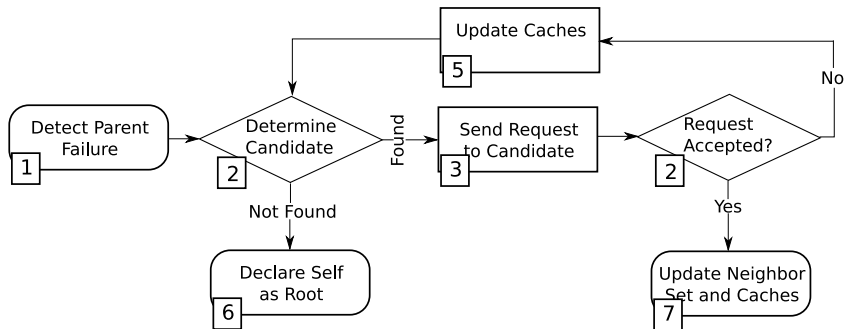
How Can We Make Trees Robust?

Approach

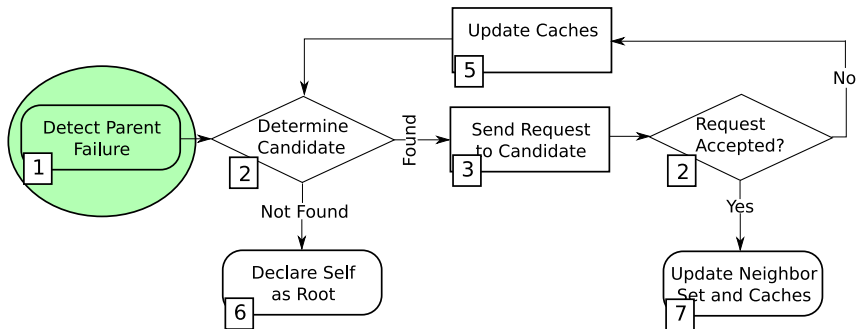
- Arrange nodes in rooted tree
- Have nodes react to parent failures
- Set of strategies and caches to locate new parents
- Cycle avoidance policy



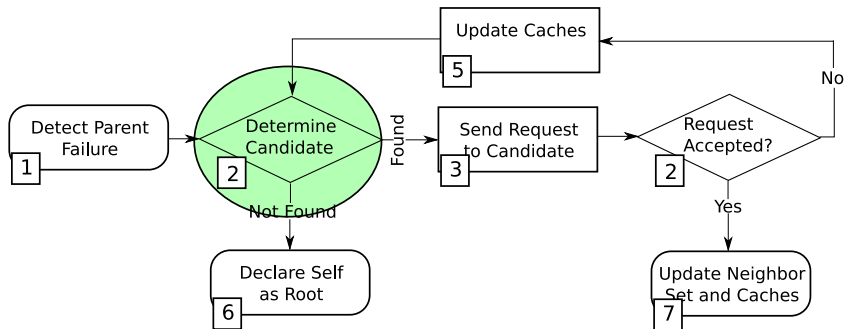
Protocol Operation



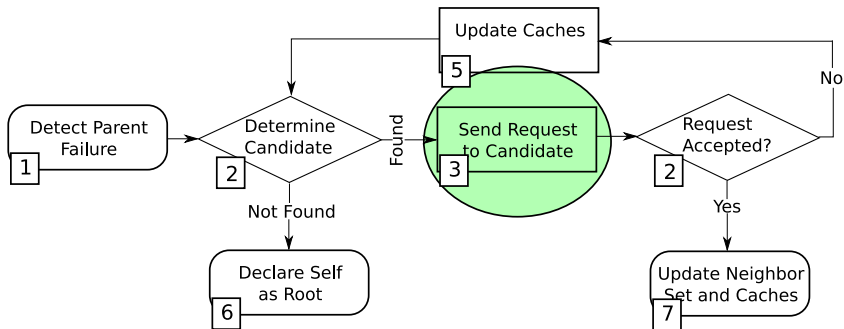
Protocol Operation



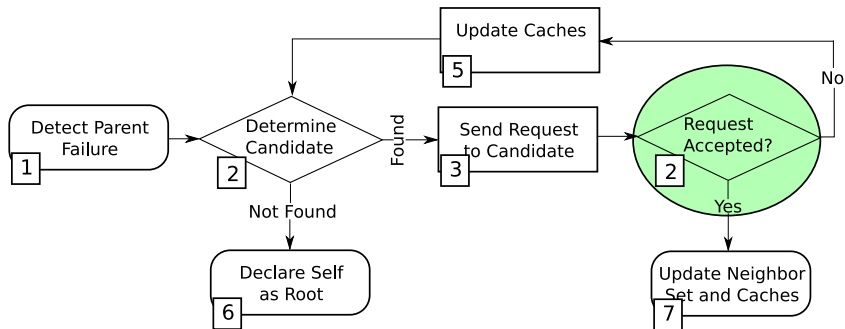
Protocol Operation



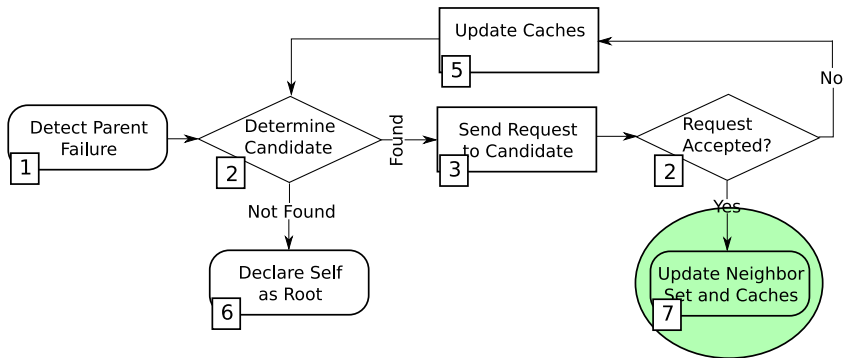
Protocol Operation



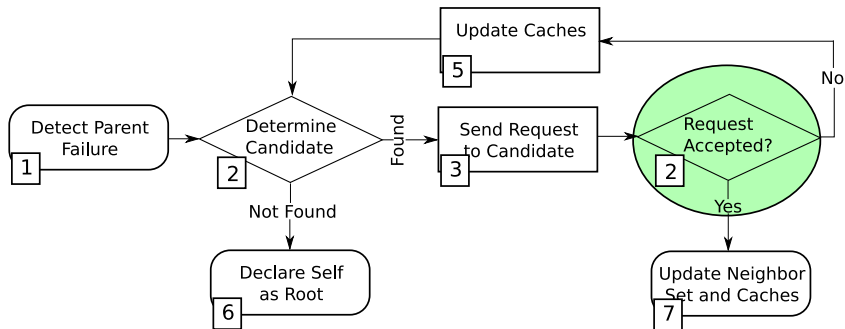
Protocol Operation



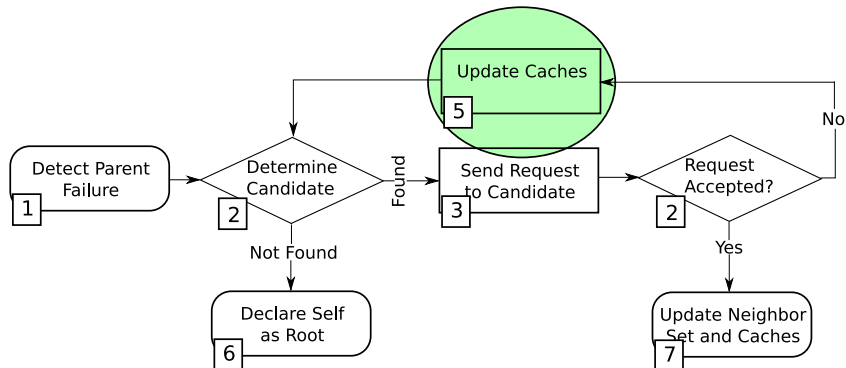
Protocol Operation



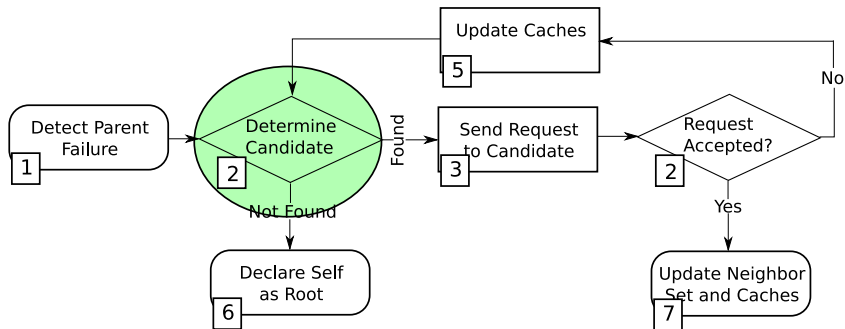
Protocol Operation



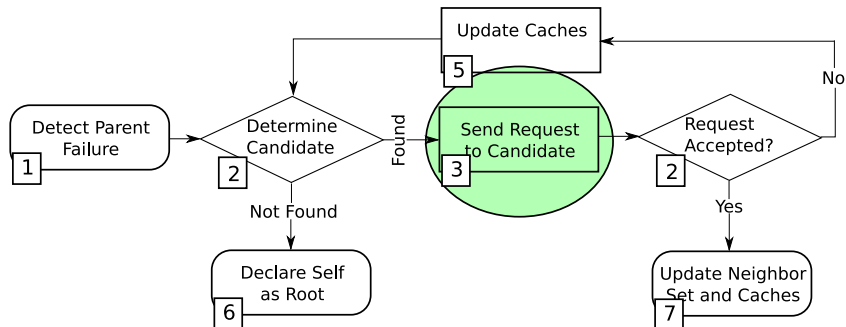
Protocol Operation



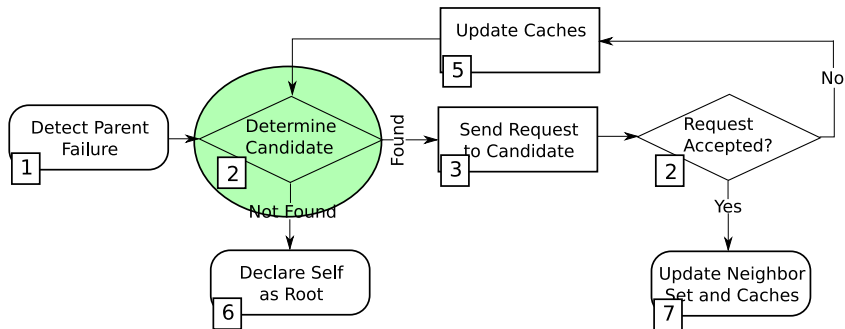
Protocol Operation



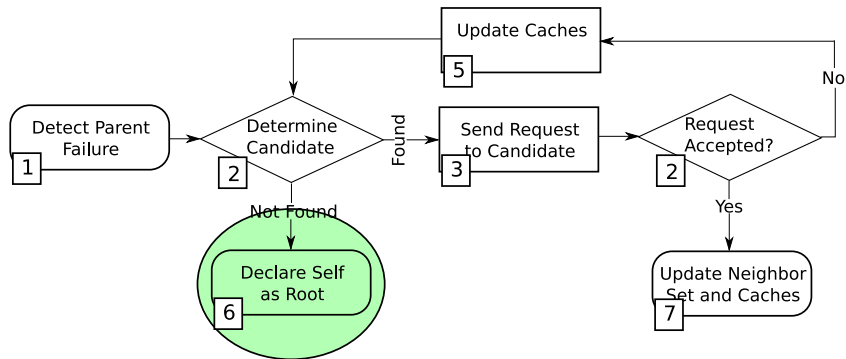
Protocol Operation



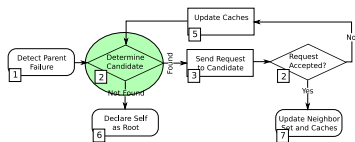
Protocol Operation



Protocol Operation



Identifying Candidate Parents



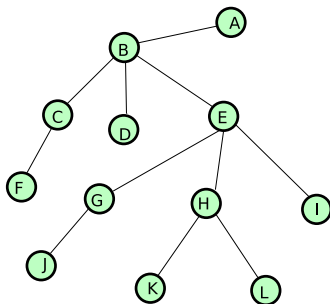
Set of Repair Strategies

- Regional
- Downstream
- BreakMaxDegree
- Upstream
- BreakMinDegree
- Global

Localizing Repairs: Regional Strategies

Regional = Ancestor Chain \cup Sibling Set

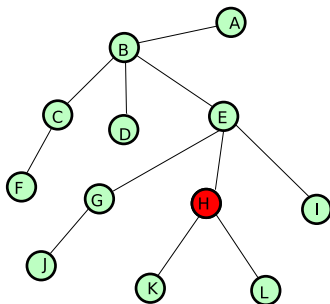
- Attempts to find a new parent close to the failed node
- Minimizes topology changes



Localizing Repairs: Regional Strategies

Regional = Ancestor Chain \cup Sibling Set

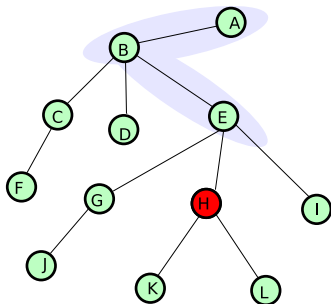
- Attempts to find a new parent close to the failed node
- Minimizes topology changes



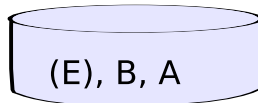
Localizing Repairs: Regional Strategies

Regional = Ancestor Chain \cup Sibling Set

- Attempts to find a new parent close to the failed node
- Minimizes topology changes



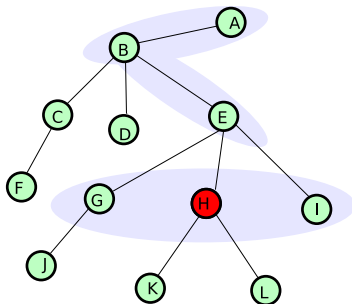
Ancestor Chain



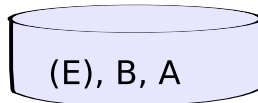
Localizing Repairs: Regional Strategies

Regional = Ancestor Chain \cup Sibling Set

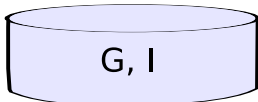
- Attempts to find a new parent close to the failed node
- Minimizes topology changes



Ancestor Chain



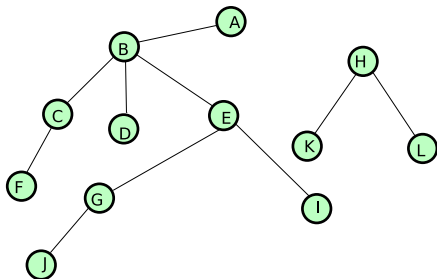
Sibling Set



Limiting Node Degree

Downstream Strategy: reach nodes with low-enough degree

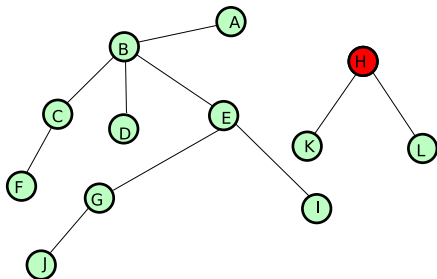
- Refuse request if own degree larger than MaxDegree
- Provide requester with references to own children
- Requester fills Downstream cache with such entries



Limiting Node Degree

Downstream Strategy: reach nodes with low-enough degree

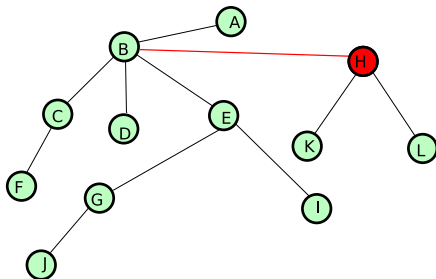
- Refuse request if own degree larger than MaxDegree
- Provide requester with references to own children
- Requester fills Downstream cache with such entries



Limiting Node Degree

Downstream Strategy: reach nodes with low-enough degree

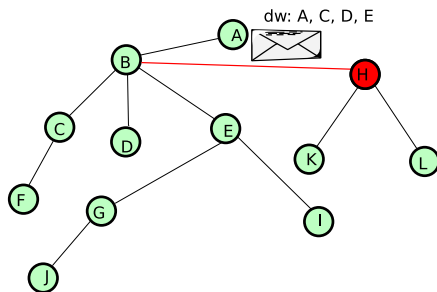
- Refuse request if own degree larger than MaxDegree
- Provide requester with references to own children
- Requester fills Downstream cache with such entries



Limiting Node Degree

Downstream Strategy: reach nodes with low-enough degree

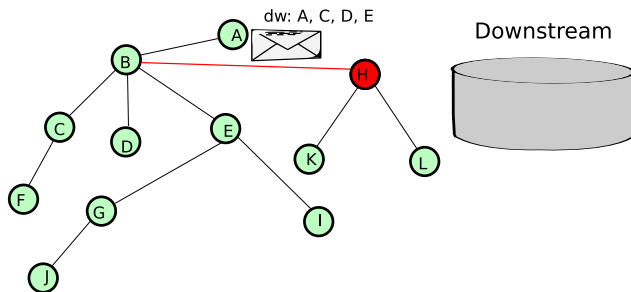
- Refuse request if own degree larger than MaxDegree
- Provide requester with references to own children
- Requester fills Downstream cache with such entries



Limiting Node Degree

Downstream Strategy: reach nodes with low-enough degree

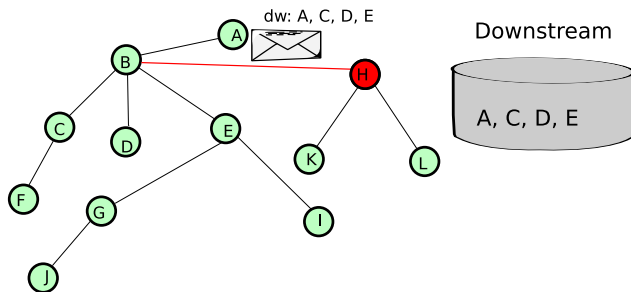
- Refuse request if own degree larger than MaxDegree
- Provide requester with references to own children
- Requester fills Downstream cache with such entries



Limiting Node Degree

Downstream Strategy: reach nodes with low-enough degree

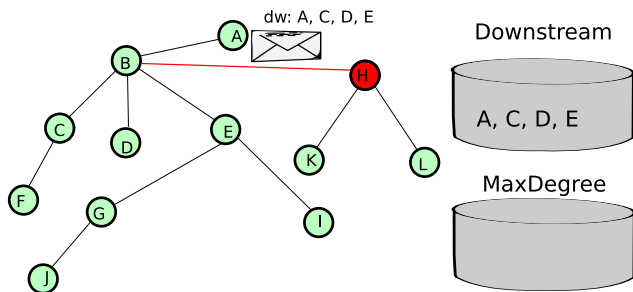
- Refuse request if own degree larger than MaxDegree
- Provide requester with references to own children
- Requester fills Downstream cache with such entries



Breaking the Degree Limit

BreakMaxDegree Strategy

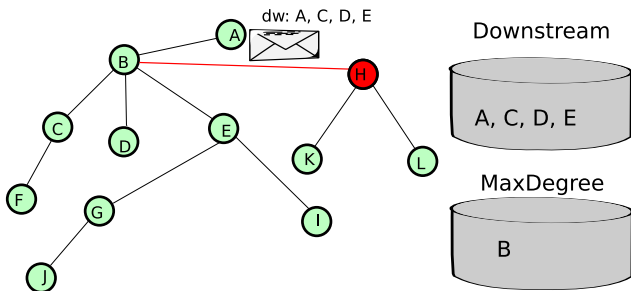
- Nodes that refuse due to MaxDegree are put in BreakMaxDegree cache.
- Nodes from BreakMaxDegree cache can be recontacted by forcing overriding of limit
- Allows for reconnection in extreme circumstances



Breaking the Degree Limit

BreakMaxDegree Strategy

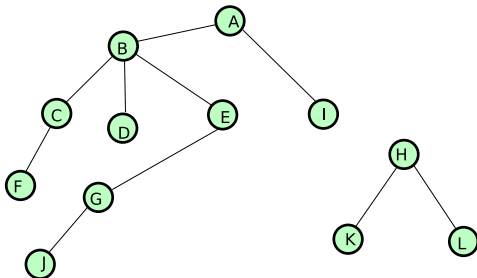
- Nodes that refuse due to MaxDegree are put in BreakMaxDegree cache.
- Nodes from BreakMaxDegree cache can be recontacted by forcing overriding of limit
- Allows for reconnection in extreme circumstances



Avoiding Line Configurations

Upstream and BreakMinDegree Strategies

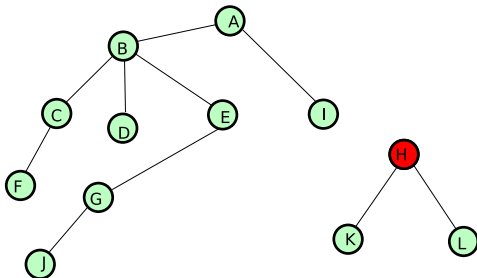
- Leaf nodes can refuse requests and provide references to upstream nodes.
- Upstream nodes placed into Upstream cache
- Refusing node placed into BreakMinDegree strategy and may be recontacted with force flag.



Avoiding Line Configurations

Upstream and BreakMinDegree Strategies

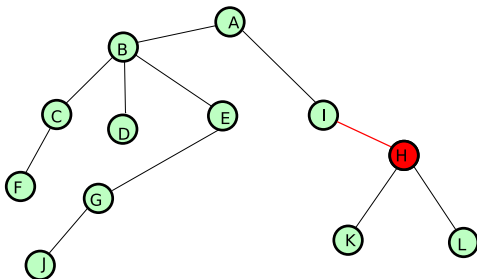
- Leaf nodes can refuse requests and provide references to upstream nodes.
- Upstream nodes placed into Upstream cache
- Refusing node placed into BreakMinDegree strategy and may be recontacted with force flag.



Avoiding Line Configurations

Upstream and BreakMinDegree Strategies

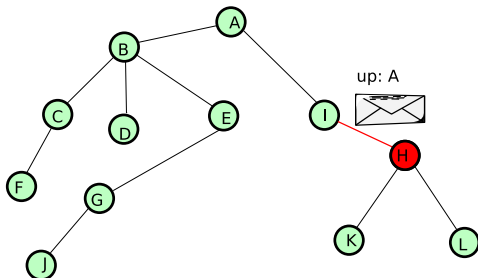
- Leaf nodes can refuse requests and provide references to upstream nodes.
- Upstream nodes placed into Upstream cache
- Refusing node placed into BreakMinDegree strategy and may be recontacted with force flag.



Avoiding Line Configurations

Upstream and BreakMinDegree Strategies

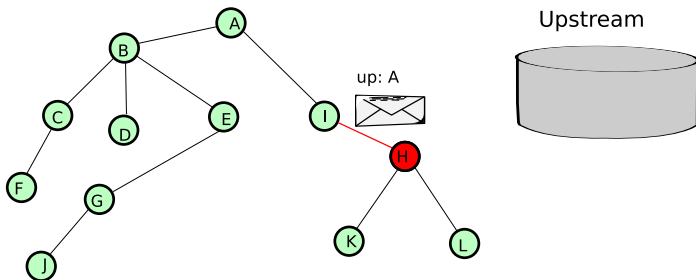
- Leaf nodes can refuse requests and provide references to upstream nodes.
- Upstream nodes placed into Upstream cache
- Refusing node placed into BreakMinDegree strategy and may be recontacted with force flag.



Avoiding Line Configurations

Upstream and BreakMinDegree Strategies

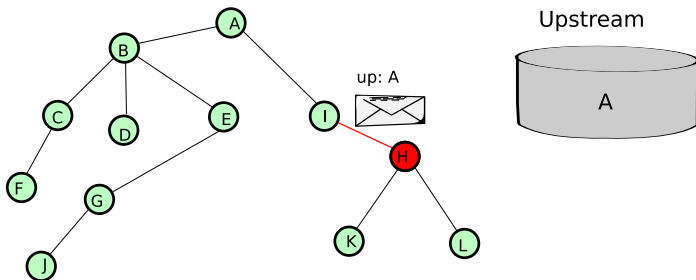
- Leaf nodes can refuse requests and provide references to upstream nodes.
- Upstream nodes placed into Upstream cache
- Refusing node placed into BreakMinDegree strategy and may be recontacted with force flag.



Avoiding Line Configurations

Upstream and BreakMinDegree Strategies

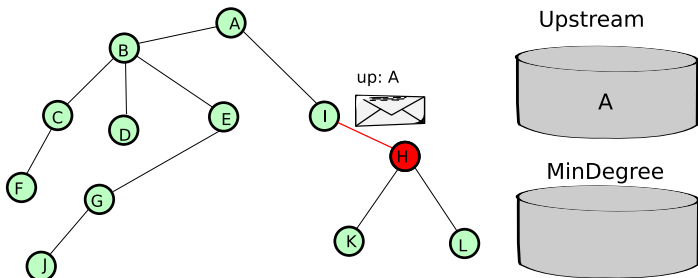
- Leaf nodes can refuse requests and provide references to upstream nodes.
- Upstream nodes placed into Upstream cache
- Refusing node placed into BreakMinDegree strategy and may be recontacted with force flag.



Avoiding Line Configurations

Upstream and BreakMinDegree Strategies

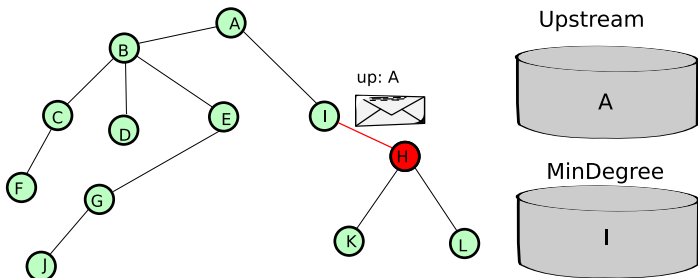
- Leaf nodes can refuse requests and provide references to upstream nodes.
- Upstream nodes placed into Upstream cache
- Refusing node placed into BreakMinDegree strategy and may be recontacted with force flag.



Avoiding Line Configurations

Upstream and BreakMinDegree Strategies

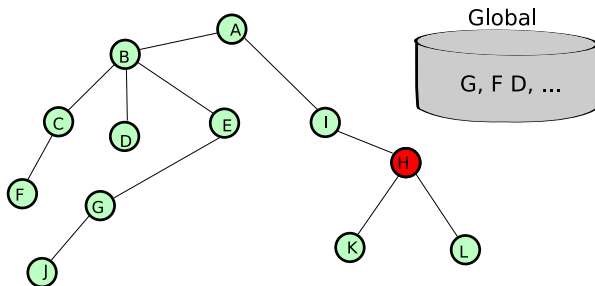
- Leaf nodes can refuse requests and provide references to upstream nodes.
- Upstream nodes placed into Upstream cache
- Refusing node placed into BreakMinDegree strategy and may be recontacted with force flag.



Managing catastrophes

Global cache

- Contains references to random nodes from anywhere in the network
- May be maintained through a random peer-sampling layer
- Essential for bootstrapping and reconnecting partitions

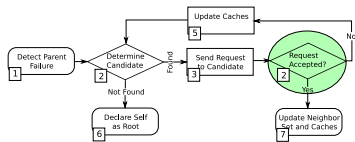


Putting the Strategies Together

Assign Priority to strategies

- Choose highest priority strategy that has available candidate
- Priority sequence determines properties of repair
 - RMG: Regional, BreakMaxDegree, Global
 - RDGM: Regional, Downstream, Global, BreakMaxDegree
 - DRGM: Downstream, Regional, Global, BreakMaxDegree
 - RUmDGM: Regional, Upstream, BreakMinDegree, Downstream Global, BreakMaxDegree
 - DUmRGM: Downstream, Upstream, BreakMinDegree, Regional, Global, BreakMaxDegree

Accepting Connection Requests



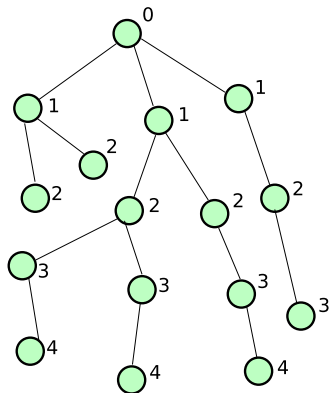
Accepting Requests

- MaxDegree Limit
- MinDegree Limit
- *Ensure Cycle Freedom*

Cycle prevention

Naive approach

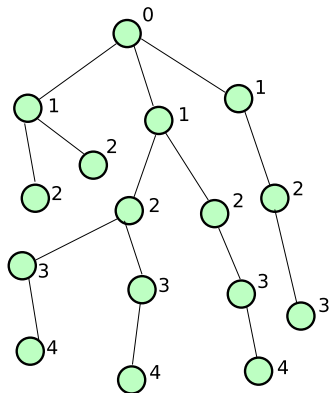
- Use hopcount from the root
- Parent accept new child if child has larger hopcount



Cycle prevention

Naive approach

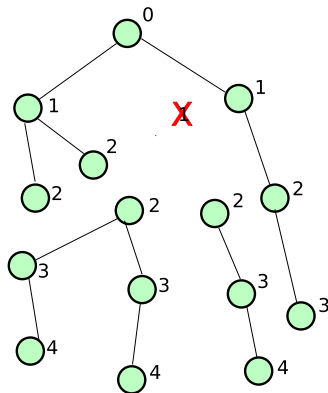
- Use hopcount from the root
- Parent accept new child if child has larger hopcount



Cycle prevention

Naive approach

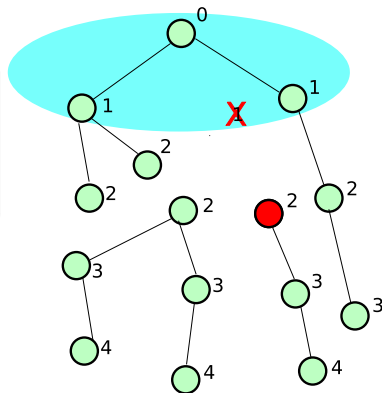
- Use hopcount from the root
- Parent accept new child if child has larger hopcount



Cycle prevention

Naive approach

- Use hopcount from the root
- Parent accept new child if child has larger hopcount

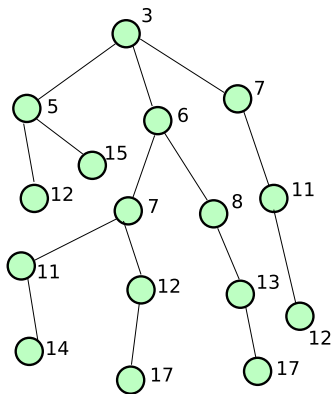


Real Valued Depth

Associate each node with real number

- Not limited to “plus-one” relationship
- Nodes can decrease their own depth to accept connection requests
- Only require child depth be *greater-than* its parent's at *all times*

Open new possibilities for reconnection

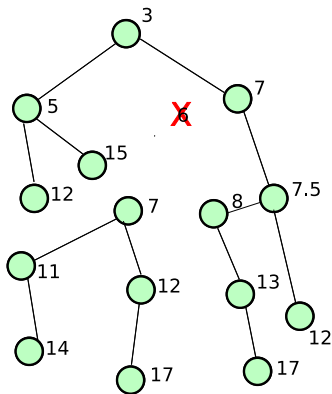


Real Valued Depth

Associate each node with real number

- Not limited to “plus-one” relationship
- Nodes can decrease their own depth to accept connection requests
- Only require child depth be *greater-than* its parent's at *all times*

Open new possibilities for reconnection



Tree Identifier

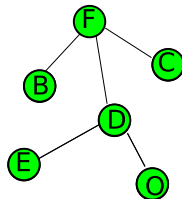
Depth allows trees to be repaired

Analogous mechanism needed for tree merging

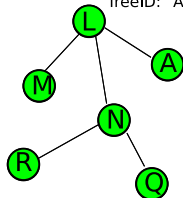
- to recover from partitions
- to merge separate overlays

- Total ordering between trees
- Partition -> append new root's id to old tree-identifier
- Used in combination with depth to guarantee cycle freedom

TreeID: "CF"



TreeID: "AL"



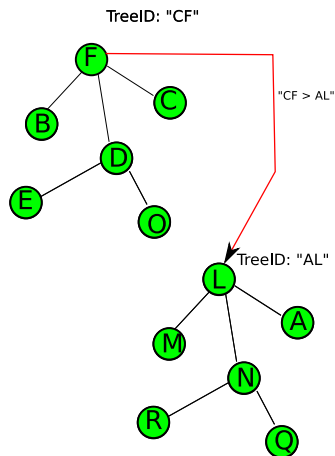
Tree Identifier

Depth allows trees to be repaired

Analogous mechanism needed for tree merging

- to recover from partitions
- to merge separate overlays

- Total ordering between trees
- Partition -> append new root's id to old tree-identifier
- Used in combination with depth to guarantee cycle freedom



Tree Identifier

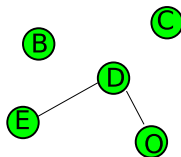
Depth allows trees to be repaired

Analogous mechanism needed for tree merging

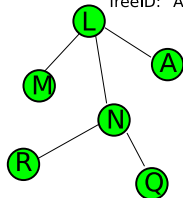
- to recover from partitions
- to merge separate overlays

- Total ordering between trees
- Partition -> append new root's id to old tree-identifier
- Used in combination with depth to guarantee cycle freedom

TreeID: "CF"



TreeID: "AL"



Tree Identifier

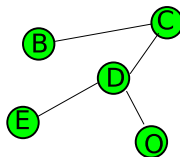
Depth allows trees to be repaired

Analogous mechanism needed for tree merging

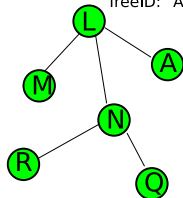
- to recover from partitions
- to merge separate overlays

- Total ordering between trees
- Partition -> append new root's id to old tree-identifier
- Used in combination with depth to guarantee cycle freedom

TreeID: "CFC"



TreeID: "AL"



Simulation Results

Setting

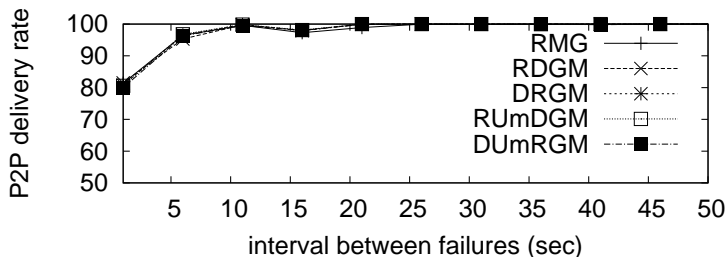
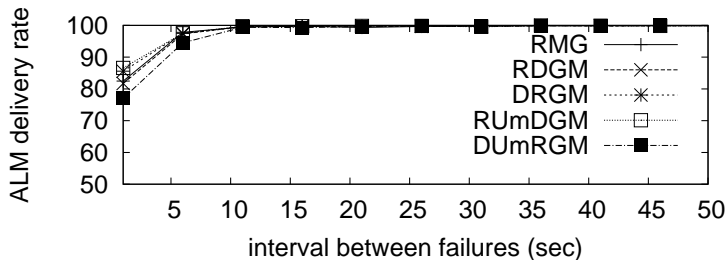
- Gnutella Trace + Varying Degrees of Catastrophic Failures
- Two Applications
 - *MCAST*: one message every 10 sec
 - *P2P*: one message per node every 100 sec

Protocol Instances

- **RMG**: Regional, BreakMaxDegree, Global
- **RDGM**: Regional, Downstream, Global, BreakMaxDegree
- **DRGM**: Downstream, Regional, Global, BreakMaxDegree
- **RUmDGM**: Regional, Upstream, BreakMinDegree, Downstream Global, BreakMaxDegree
- **DUmRGM**: Downstream, Upstream, BreakMinDegree, Regional, Global, BreakMaxDegree

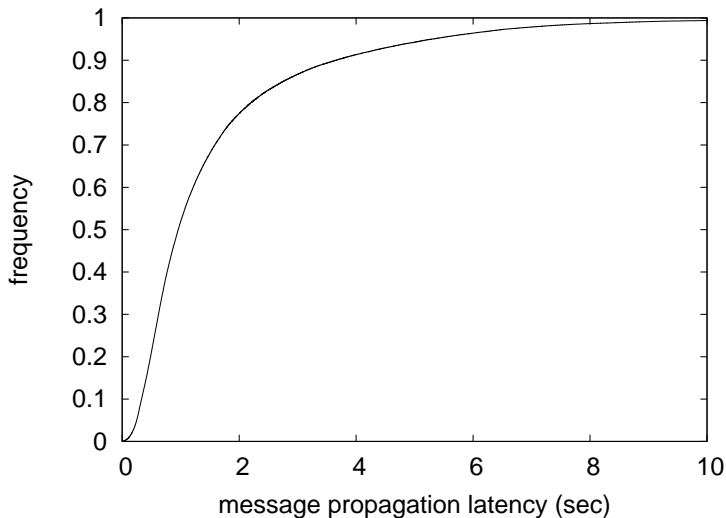
Delivery Rates

Network of 1500 nodes

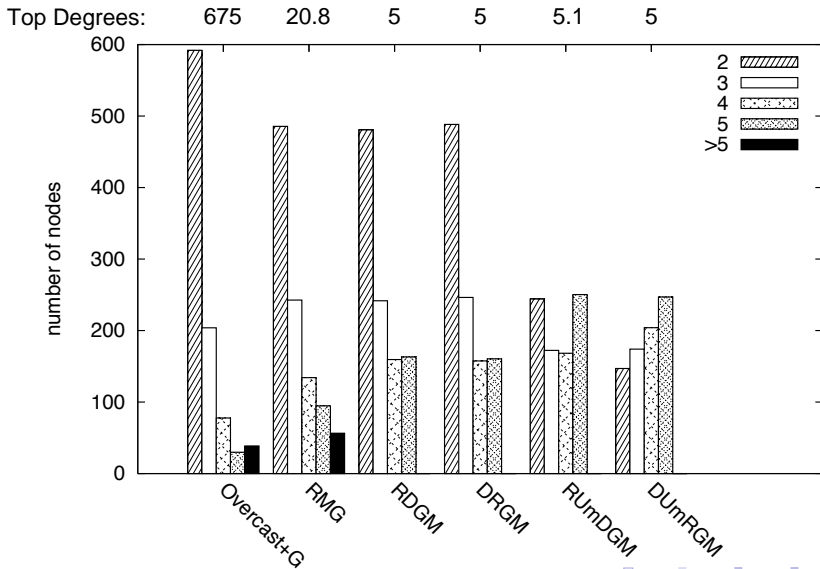


Message Latency: PlanetLab

Dominated by connection-establishment and processing delays

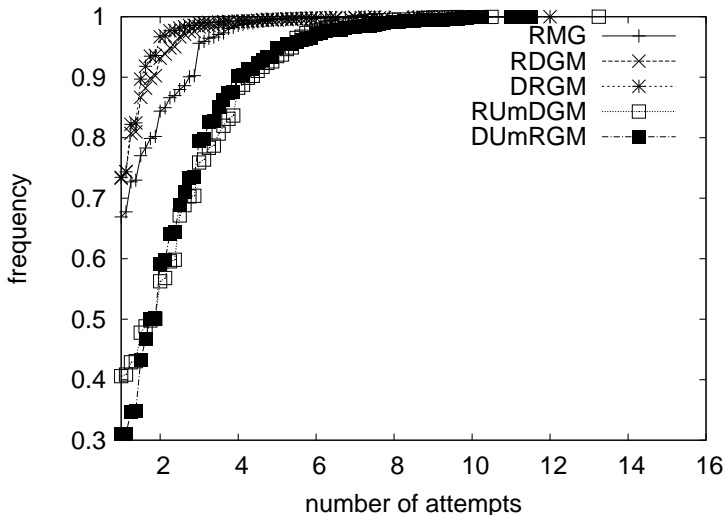


Node Degree



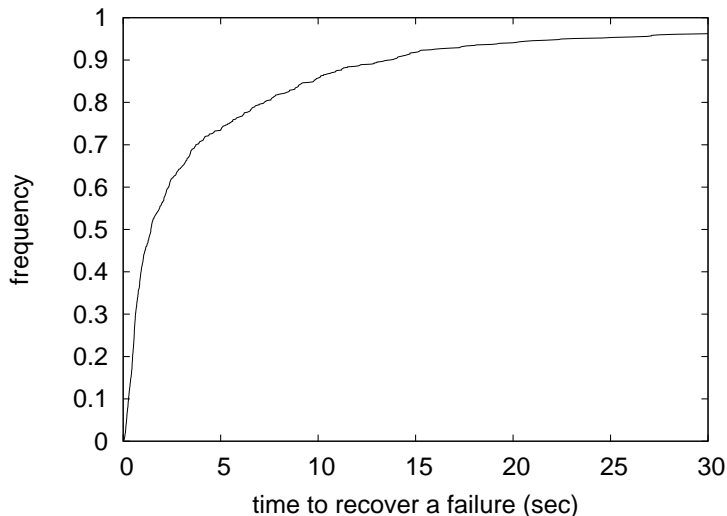
Repair Latency

Candidates contacted per repair: cumulative distribution



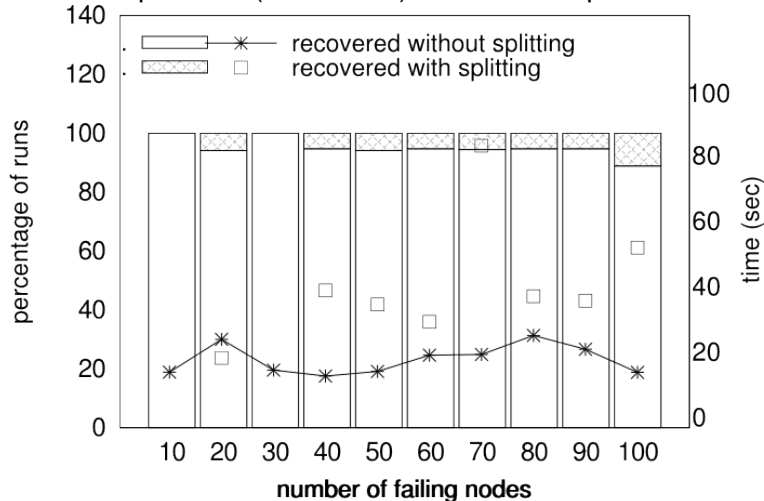
Repair Latency: PlanetLab

Connect to new parent with random, uncorrelated failures



Repair Latency: PlanetLab

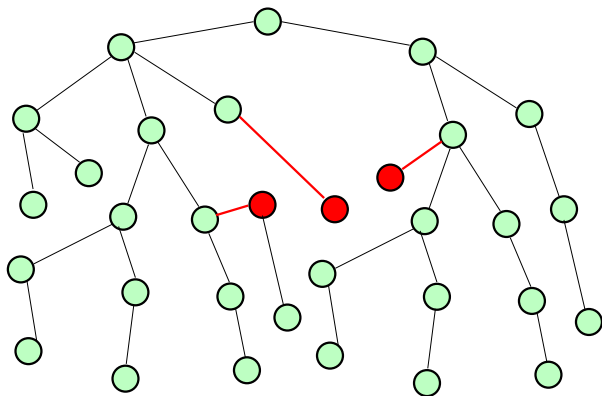
Time to repair tree (130 nodes) after catastrophic failure



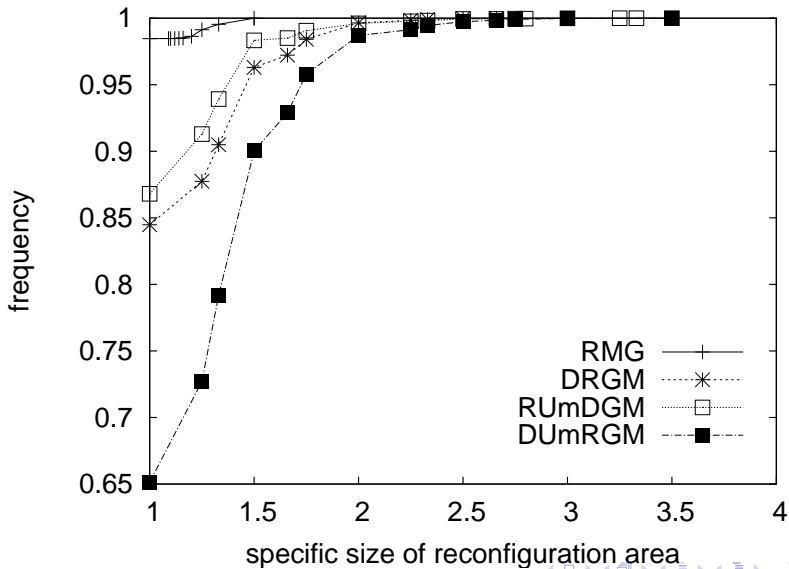
Reconfiguration Area

Measure of area affected by topology change

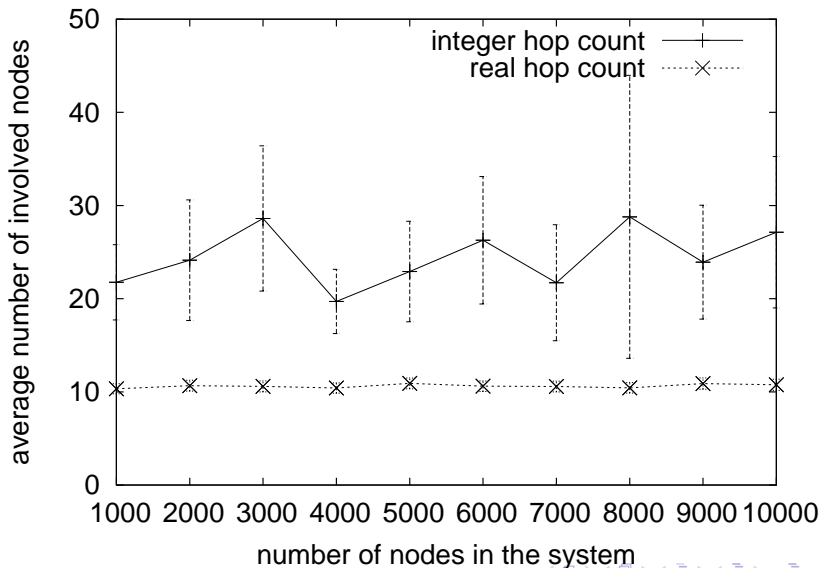
Represents affected nodes in tree-based content-based publish-subscribe.



Recovery Locality



Real-Valued vs Integer-HopCount Depth



Summary

Trees ain't dead in P2P

- Made trees robust in dynamic environment
- Manage properties of resulting topology
- Key contributions:
 - Real-valued node-depth
 - Repair Strategies and combinations
 - Impact of strategies on topology

Going Beyond

- Evaluate on more realistic applications
- Parallelize requests to improve latency
- Tree-vs-Mesh break-even point

Thank you!

